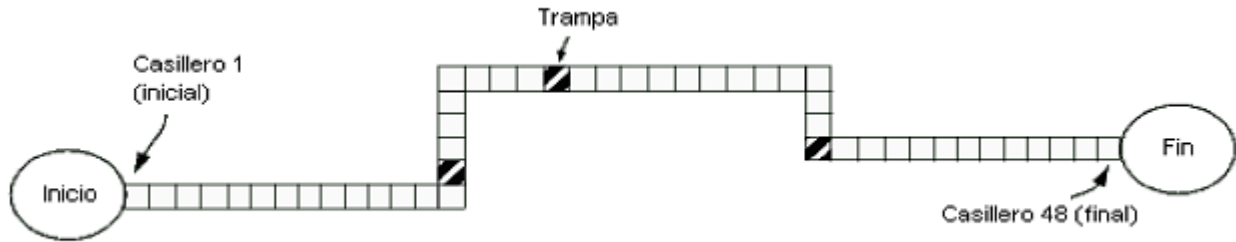


Julio Javier Castillo, Diego Javier Serrano
*Laboratorio de Investigación de Software MsLabs, Dpto. Ing. en
 Sistemas de Información, Facultad Regional Córdoba - Universidad
 Tecnológica Nacional (Argentina)*

<jotacastillo@gmail.com>,
 <diegojserrano@gmail.com>

El enunciado de este problema apareció en el número 205 de *Novática* (mayo-junio 2010, p. 76). Es el problema D de los planteados en la Primera Competencia de Programación de la Facultad Regional de Córdoba (Universidad Tecnológica Nacional, Argentina) UTN-FRC celebrada en octubre de 2009.



Presentación del problema

En este problema se nos solicita programar un juego de tablero muy simple, el cual posee algunas similitudes con juegos tradicionales como el Ludo o la Oca.

La solución es básicamente trivial, por cada jugador necesitamos almacenar únicamente la posición en que se encuentra en cada momento durante el juego y cada vez que lance los dados, incrementaremos esa posición según el valor del lanzamiento. Cuando el valor de la posición de algún jugador supere el último casillero, el juego finaliza con dicho jugador como ganador.

Aún cuando la cantidad de jugadores es limitada, la opción más práctica es almacenar las posiciones de los jugadores en un arreglo, asignando cada posición del arreglo a un jugador, en el orden de lanzamiento de los dados, desde 0 hasta p-1 (con p igual a la cantidad de jugadores).

19	25	14	20
0	1	2	3

Durante la lectura de los lanzamientos de los dados, mantendremos una variable que indique a qué jugador corresponde el próximo lanzamiento. Esa variable se incrementará en uno cada vez que se lee un nuevo lanzamiento y se la utilizará para hacer acceso directo al arreglo de posiciones. La variable se reiniciará a 0 cuando alcance el valor de p. De esa forma logramos recorrer el arreglo en forma circular, iniciando desde el inicio (posición 0) cuando alcanzamos el final (posición p-1)

Trampas

Sin embargo el juego plantea una dificultad adicional. Tres casilleros del tablero se encuentran marcados como "trampas". Cuando un jugador finaliza su turno en una trampa, pierde su próximo turno. Es decir que la próxima vez que deba lanzar los dados, no lo hace, quedando en el mismo casillero y continuando con el lanzamiento de dados para el jugador siguiente.

Pero en los datos de entrada no se nos indica de ninguna forma a qué jugador corresponde cada jugada, por lo tanto resulta de especial

cuidado nuestra tarea de asignar los lanzamientos a cada jugador, "saltando" a los jugadores que caigan en trampas.

Para ello vamos a mantener otro arreglo que contendrá una variable booleana indicando si el jugador está habilitado para jugar. Este segundo arreglo será paralelo al de posiciones, es decir que ambos serán accedidos por el mismo índice y que cada posición corresponderá al mismo jugador en los dos arreglos.

El arreglo de jugadores habilitados iniciará con todas sus posiciones con valor `true`, y se asignarán a `false` a medida que los jugadores caigan en las trampas. Luego de que un jugador juegue, para identificar a quién corresponde el próximo lanzamiento de dados, se recorrerá el arreglo de jugadores habilitados desde la posición del último jugador hasta encontrar un casillero con valor `true`. A medida que encontremos casilleros con `false`, deberán ser asignados a `true`, ya que cada jugador es inhabilitado por un único turno.

true	true	false	true
0	1	2	3

A jugar

El problema nos garantiza que la cantidad de lanzamientos alcanza exactamente para encontrar un ganador. Por lo tanto podemos analizar los lanzamientos hasta encontrar un ganador o simplemente recorrer con un ciclo hasta N (N es recibido en la entrada y su valor es la cantidad de lanzamientos de dados).

Por cada lanzamiento identificaremos a qué jugador corresponden los dados lanzados e incrementaremos el valor correspondiente en el arreglo de posiciones. Si luego de asignar la nueva posición, esta equivale a la de alguna de las trampas, se inhabilita al jugador para su próximo turno, asignando `false` en el arreglo de jugadores habilitados. Ya que todos los jugadores inician habilitados y que el primer lanzamiento siempre le es asignado al primer jugador, la variable que utilizaremos como índice para los arreglos será inicializada en 0.

```
import java.util.Scanner;

public class Datos {
public static void main(String[] args) {
//Datos de entrada
int p, s, t1, t2, t3, n, d1, d2;
// Posiciones
int []pos = new int[10];
//Jugadores habilitados
boolean []hab = new boolean[10];
//i para recorridos, j para el jugador que
lanza
int i, j;
Scanner sc = new Scanner(System.in);
p = sc.nextInt();
s = sc.nextInt();
while (p != 0)
{
for (i=0; i<p; i++)
{
pos[i] = 0;
hab[i] = true;
}
t1 = sc.nextInt();
t2 = sc.nextInt();
t3 = sc.nextInt();
n = sc.nextInt();
j = 0;
for(i=0; i<n; i++)
{
d1 = sc.nextInt();
d2 = sc.nextInt();
pos[j]+=(d1+d2);
if (pos[j] > s) break;
if (pos[j]==t1 ||pos[j]==t2 ||pos[j]==t3)
hab[j]=false;
j++;
j%=p;
while (!hab[j]){
hab[j] = true;
j++;
j%=p;
}
}
System.out.println(j+1);
p = sc.nextInt();
s = sc.nextInt();
}
}
}
```